自：https://www.cnblogs.com/Remenber-Ray/p/8858032.html

[**log4j2教程【RollingFileAppender】**](https://www.cnblogs.com/Remenber-Ray/p/8858032.html)

# 说明

**rollover 表示的是当日志文件大小满足指定大小后，就生成一个新的文件的过程。**

# RollingFileAppender

RollingFileAppender是一个OutputStreamAppender，它（会把日志）写入到filename参数命名的文件中，并且会根据TriggeringPolicy和RolloverPolicy来rollover（rolls the file over）。RollingFileAppender会使用RollingFileManager（继承OutputStreamManager）来实际执行文件的I/O和执行rollover。尽管不能共享来做不同配置的RolloverFileAppenders，但是如果Manager可以访问的话，那么RolloverFileAppenders可以（进行共享）。   
例如：在一个servlet容器中的两个web应用程序，他们有自己的配置，如果他们Log4j是共用一个类加载器（ClassLoader）那么就可以安全的写入到一个文件中。

RollingFileAppender 需要TriggeringPolicy和RolloverStrategy。triggering policy决定是否应该执行rollover的操作，而RolloverStrategy定义了应该如何完成rollover。如果RolloverStrategy没有配置的话，RollingFileAppender将使用DefaultRolloverStrategy。从log4j2.5版本开始，在DefaultRolloverStrategy中配置的自定义删除操作在rollover时将被执行。   
从2.8版本开始，如果在DirectWriteRolloverStrategy中没有配置文件名，将使用DefaultRolloverStrategy进行替换。

RollingFileAppender不支持文件锁的。

**这里rollover的操作可以理解为：当日志文件大小满足指定大小后，就生成一个新的文件。**

# RollingFileAppender参数

| **参数名** | **类型** | **描述** |
| --- | --- | --- |
| append | boolean | 默认为true，记录追加到文件的最后，否则就先清除以前的记录再写入 |
| bufferedIO | boolean | 默认true,记录将会写入到缓存区，当缓存区满的时候，就会写入磁盘。或者如果设置immediateFlush将会立即写入。文件锁定不能和bufferedIO一起使用。 |
| bufferSize | int | 当bufferedIO设置为true是，默认是8192 bytes |
| createOnDemand | boolean | 默认为false，该appender按需创建文件，当日志事件通过所有的filters并且通过路由指向了该appender,该appender仅仅创建该文件 |
| filter | Filter | 过滤器决定事件是否应该由这个Appender来处理。通过使用CompositeFilter来使用多个Filter |
| fileName | String | 要写入的文件的名称。如果文件或其父目录不存在，它们都将被创建出来 |
| filePattern | String | 压缩日志文件的文件名的模式。该模式的格式取决于所使用的RolloverPolicy。DefaultRolloverPolicy将接受兼容SimpleDateFormat的日期/时间模式和/或者%i(代表整数计数器)。这个模式也支持运行时插值，所以任何的查找( eg:[DateLookup](http://logging.apache.org/log4j/2.x/manual/lookups.html#DateLookup))都可以包含在模式中 |
| immediateFlush | boolean | 默认为true，每次写入都会执行flush。这可以保证每次数据都被写入磁盘，但是会影响性能。在同步的loggers中每次写入执行flush，那就非常有用。异步loggers和appenders将会在一系列事件结束后自动执行flush，即使设置为false。这也保证了数据写入到磁盘而且很高效 |
| layout | Layout | 这个Layout用于格式化LogEvent.如果没有提供默认的layout，默认为layout模式为%m%n。 |
| name | String | 该Appender名称 |
| policy | TriggeringPolicy | 用于决定是否发生rollover的策略 |
| strategy | RolloverStrategy | 用于决定压缩文件的名称和路径 |
| ignoreExceptions | boolean | 默认为true，遇到异常时，会将事件追加到内部日志并忽略它。设置false时，异常会传递给调用者，当这个appender被FailoverAppender包裹时，必须设置为false |

## Triggering Policies

### Composite Triggering Policy

Composite Triggering Policy组合了多个triggering policies，如果配置的策略中的任何一个返回true，则返回true。CompositeTriggeringPolicy简单的通过在policies元素包裹其他的policies来配置。   
例如，以下XML片段定义了当JVM启动时，当日志大小达到二十兆字节以及当前日期与日志的开始日期不匹配时滚动日志的策略。

<Policies> <OnStartupTriggeringPolicy /> <SizeBasedTriggeringPolicy size="20 MB" /> <TimeBasedTriggeringPolicy /> </Policies>

### Cron Triggering Policy

基于cron表达式的CronTriggeringPolicy触发rollover。

CronTriggeringPolicy 参数

| **参数名称** | **类型** | **描述** |
| --- | --- | --- |
| schedule | String | cron表达式。该表达式和Quartz调度所允许的表达式相同，有关表达式的完整描述，请参阅[CronExpression](http://logging.apache.org/log4j/2.x/log4j-core/apidocs/org/apache/logging/log4j/core/util/CronExpression.html) |
| evaluateOnStartup | boolean | 在启动时，将根据文件的最后修改时间戳评估cron表达式。如果cron表达式表示在该时间和当前时间之间应该发生rollover，则文件将立即rollover。 |

### OnStartup Triggering Policy

如果日志文件比当前jvm启动时间更早以及满足或者超过最小文件的大小就会触发rollover

OnStartupTriggeringPolicy 参数说明:

| **参数名称** | **类型** | **描述** |
| --- | --- | --- |
| minSize | long | 文件必定发生rollover操作的最小尺寸。要是大小为0的话，那么无论文件大小是多少都将引起rollover。默认值为1，这将阻止空文件发送rollover |

### SizeBased Triggering Policy

SizeBased Triggering Policy：一旦文件大小达到指定大小后，就会发送rollover。   
该策略接受一个interval属性和modulate布尔属性。其中interval属性表示的是，基于时间模式应该发送rollover的频率。

TimeBasedTriggeringPolicy参数说明：

| **参数名称** | **类型** | **描述** |
| --- | --- | --- |
| interval | integer | 根据日期格式中最具体的时间单位来决定应该多久发生一次rollover。例如，在日期模式中小时为具体的时间单位，那么每4小时会发生4次rollover，默认值为1 |
| modulate | boolean | 表示是否调整时间间隔以使在时间间隔边界发生下一个rollover。例如：假设小时为具体的时间单元，当前时间为上午3点，时间间隔为4，第一次发送rollover是在上午4点，接下来是上午8点，接着是中午，接着是下午4点等发生。 |

## Rollover Strategies

### Default Rollover Strategy

默认的rollover strategy接受一个日期/时间模式和一个整数，其中这个整数，是RollingFileAppender本身指定的filePattern属性。如果date/time模式存在的话，它将会替换当前日期和时间的值。如果这个模式包含整数的话，它将会在每次发生rollover时，进行递增。如果模式同时包含date/time和整数，那么在模式中，整数会递增直到结果中的data/time模式发生改变。如果文件模式是以".gz", ".zip", ".bz2", ".deflate", ".pack200", or ".xz"结尾的，将会与后缀相匹配的压缩方案进行压缩文件。格式为：bzip2, Deflate, Pack200 和 XZ需要[Apache Commons Compress](http://commons.apache.org/proper/commons-compress/)，此外，XZ需要[XZ for Java](http://tukaani.org/xz/java.html)   
该模式还可以包含可以在运行时解析的查找引用，如下面的示例所示:   
默认的rollover策略支持三种增加计数的方式。第一种叫做：fixed window策略。为了说明它的工作原理，假设min属性设置为1，max属性设置为3，文件名为“foo.log”，文件名模式为：foo-％i.log。

| **rollover次数** | **输出的目标** | **压缩的日志文件** | **描述** |
| --- | --- | --- | --- |
| 0 | foo.log | - | 所有的日志都输出到初始文件中 |
| 1 | foo.log | foo-1.log | 在第一次rollover时，foo.log会被重命名为foo-1.log。同时会创建一个新的foo.log并开始写入。 |
| 2 | foo.log | foo-1.log, foo-2.log | 在第二次发生rollover时，foo-1.log会重命名为foo-2.log并且foo.log会重命名为foo-1.log。同时会创建一个新的foo.log并开始写入。 |
| 3 | foo.log | foo-1.log, foo-2.log, foo-3.log | 在第三次发生rollover时，foo-2.log会重命名为foo-3.log。foo-1.log重命名为foo-2.log,foo.log会重命名为foo-1.log。同时会创建一个新的foo.log并开始写入。 |
| 4 | foo.log | foo-1.log, foo-2.log, foo-3.log | 在第四次和随后的rollover时，foo-3.log会被删除，foo-2.log重命名为foo-3.log。foo-1.log重命名为foo-2.log。foo.log重命名为foo-1.log。后面同理 |

相比之下，当 fileIndex属性设置了max,其他设置和上面相同，将执行以下操作：

| **rollover次数** | **输出的目标** | **压缩的日志文件** | **描述** |
| --- | --- | --- | --- |
| 0 | foo.log | - | 所有的日志都输出到初始文件中 |
| 1 | foo.log | foo-1.log | 在第一次rollover时，foo.log会被重命名为foo-1.log。同时会创建一个新的foo.log并开始写入。 |
| 2 | foo.log | foo-1.log foo-2.log | 在第二次rollover时，foo.log重命名为foo-2.log。同时会创建一个新的foo.log并开始写入。 |
| 3 | foo.log | foo-1.log foo-2.log foo-3.log | 在第三次发生rollover时，fool.log重命名为foo-3.log,同时会创建一个新的foo.log并开始写入。 |
| 4 | foo.log | foo-1.log foo-2.log foo-3.log | 在第四次和随后的rollover时，foo-1.log会被删除，foo-2.log重命名为foo-1.log,foo-3.log重命名为foo-2.log，foo.log重命名为foo-3.log。同时会创建一个新的foo.log并开始写入。 |

最后，从2.8版本开始，如果fileIndex属性设置为nomax，那么最大和最小值，都将会被忽略掉，文件编号将从1开发增加，并且每次rollover时递增都从编码最大开始（项目于max效果），而且没有文件数的限制。

DefaultRolloverStrategy参数

| **参数名称** | **类型** | **描述** |
| --- | --- | --- |
| fileIndex | String | 如果设置了max(默认就是），文件索引（编号）高的比低的更 新些。如果设置min，文件重命名将遵循Fixed Window策略 |
| min | integer | 计数器的最小值。默认值为1。 |
| max | integer | 计数器的最大值。一旦达到这个值，旧的档案将在随后的rollover中被删除。 |
| compressionLevel | integer | 设置压缩级别0-9，其中0=无，1=最佳速度，通过9=最佳压缩。只适用于ZIP文件。 |

### DirectWrite Rollover Strategy

DirectWriteRolloverStrategy会将日志事件会直接写入文件模式表示的文件中去。使用这个策略不会执行文件重命名。如果基于大小的触发策略导致在指定的时间段内写入多个文件，则它们将从一个编号开始，并持续递增，直到发生基于时间的rollover。   
警告：如果文件模式里有压缩的后缀，那么当应用程序关闭时，当前文件将不被压缩。此外，如果时间变化使得文件模式不再匹配当前文件，则它也不会在启动时被压缩。

DirectWriteRolloverStrategy 参数

| **参数名称** | **类型** | **描述** |
| --- | --- | --- |
| maxFiles | String | 在与文件模式（file pattern）匹配的时间段内允许的最大文件数。如果这个数字被突破了，则最旧的文件将被删除。如果指定了，那么这个值必须大于1。如果值小于零或省略，则文件数不受限制。 |
| compressionLevel | integer | 设置压缩级别0-9，其中0=无，1=最佳速度，通过9=最佳压缩。只适用于ZIP文件。 |

①下面是使用RollingFileAppender的一个示例配置，并且是基于时间和大小的触发策略。其将会根据当前年和月份在未来7天内创建7个压缩包，并且使用gzip进行压缩。

<?xml version="1.0" encoding="UTF-8"?> <Configuration status="warn" name="MyApp" packages=""> <Appenders> <RollingFile name="RollingFile" fileName="logs/app.log" filePattern="logs/$${date:yyyy-MM}/app-%d{MM-dd-yyyy}-%i.log.gz"> <PatternLayout> <Pattern>%d %p %c{1.} [%t] %m%n</Pattern> </PatternLayout> <Policies> <TimeBasedTriggeringPolicy /> <SizeBasedTriggeringPolicy size="250 MB"/> </Policies> </RollingFile> </Appenders> <Loggers> <Root level="error"> <AppenderRef ref="RollingFile"/> </Root> </Loggers> </Configuration>

②第二个例子显示一个rollover策略，最多保留20个文件：

<?xml version="1.0" encoding="UTF-8"?> <Configuration status="warn" name="MyApp" packages=""> <Appenders> <RollingFile name="RollingFile" fileName="logs/app.log" filePattern="logs/$${date:yyyy-MM}/app-%d{MM-dd-yyyy}-%i.log.gz"> <PatternLayout> <Pattern>%d %p %c{1.} [%t] %m%n</Pattern> </PatternLayout> <Policies> <TimeBasedTriggeringPolicy /> <SizeBasedTriggeringPolicy size="250 MB"/> </Policies> <DefaultRolloverStrategy max="20"/> </RollingFile> </Appenders> <Loggers> <Root level="error"> <AppenderRef ref="RollingFile"/> </Root> </Loggers> </Configuration>

③下面是使用RollingFileAppender的一个示例配置，并且是基于时间和大小的触发策略。其将会根据当前年和月份在未来7天内创建7个压缩包，并且每6小时即被6整除时，会使用gzip进行压缩每个文件。

<?xml version="1.0" encoding="UTF-8"?> <Configuration status="warn" name="MyApp" packages=""> <Appenders> <RollingFile name="RollingFile" fileName="logs/app.log" filePattern="logs/$${date:yyyy-MM}/app-%d{yyyy-MM-dd-HH}-%i.log.gz"> <PatternLayout> <Pattern>%d %p %c{1.} [%t] %m%n</Pattern> </PatternLayout> <Policies> <!--这行区别--> <TimeBasedTriggeringPolicy interval="6" modulate="true"/> <SizeBasedTriggeringPolicy size="250 MB"/> </Policies> </RollingFile> </Appenders> <Loggers> <Root level="error"> <AppenderRef ref="RollingFile"/> </Root> </Loggers> </Configuration>

④此示例配置使用基于cron和基于大小的触发策略的RollingFileAppender，并且是无限制文件数量的直接写入到归档文件中。cron触发器是每小时发生rollover并且文件大小不得超过250M。

<?xml version="1.0" encoding="UTF-8"?> <Configuration status="warn" name="MyApp" packages=""> <Appenders> <RollingFile name="RollingFile" filePattern="logs/app-%d{yyyy-MM-dd-HH}-%i.log.gz"> <PatternLayout> <Pattern>%d %p %c{1.} [%t] %m%n</Pattern> </PatternLayout> <Policies> <CronTriggeringPolicy schedule="0 0 \* \* \* ?"/> <SizeBasedTriggeringPolicy size="250 MB"/> </Policies> </RollingFile> </Appenders> <Loggers> <Root level="error"> <AppenderRef ref="RollingFile"/> </Root> </Loggers> </Configuration>

⑤此示例和上面④大致一样，但是每小时保存的文件数量限制了为10：

<?xml version="1.0" encoding="UTF-8"?> <Configuration status="warn" name="MyApp" packages=""> <Appenders> <RollingFile name="RollingFile" filePattern="logs/app-%d{yyyy-MM-dd-HH}-%i.log.gz"> <PatternLayout> <Pattern>%d %p %c{1.} [%t] %m%n</Pattern> </PatternLayout> <Policies> <CronTriggeringPolicy schedule="0 0 \* \* \* ?"/> <SizeBasedTriggeringPolicy size="250 MB"/> </Policies> <DirectWriteRolloverStrategy maxFiles="10"/> </RollingFile> </Appenders> <Loggers> <Root level="error"> <AppenderRef ref="RollingFile"/> </Root> </Loggers> </Configuration>

### Log Archive Retention Policy（日志存档保留策略）：Delete on Rollover

log4j-2.5开始引入了删除操作，使得用户更有效的的控制在rollover时间内删除文件，而不是使用DefaultRolloverStrategy max属性进行删除。删除操作允许用户配置一个或多个条件，选择要删除相对于基本目录的文件。   
注意：删除任何文件这是允许的操作。不仅仅是rollover时的文件。所以使用这个操作时，一定要小心。使用testMode参数可以测试您的配置，而不会意外删除错误的文件。

Delete 参数：

| **参 数名称** | **类型** | **描述** |
| --- | --- | --- |
| basePath | String | 必参。从哪里扫描要删除的文件的基本路径。 |
| maxDepth | int | 要访问的目录的最大级别数。值为0表示仅访问起始文件（基本路径本身），除非被安全管理者拒绝。Integer.MAX\_VALUE的值表示应该访问所有级别。默认为1，意思是指定基本目录中的文件。 |
| followLinks | boolean | 是否遵循符号链接默认值为false。 |
| testMode | boolean | 默认false。如果为true，文件将不会被删除，而是将信息打印到info级别的[status logger](http://logging.apache.org/log4j/2.x/manual/configuration.html#StatusMessages),可以利用这个来测试，配置是否和我们预期的一样 |
| pathSorter | PathSorter | 一个实现了[PathSorter](http://logging.apache.org/log4j/2.x/log4j-core/apidocs/org/apache/logging/log4j/core/appender/rolling/action/PathSorter.html)接口的插件在选择删除文件前进行排序。默认是最近修改的文件排在前面 |
| pathConditions | PathCondition[] | 如果没有指定ScriptCondition，则为必需。一个或多个PathCondition元素。如果指定了多个条件，在删除之前，他们需要接受全部的路径。条件是可以嵌套的，在这种情况下，只有在外部路径被接受的情况下，才会去评估内部路径。如果条件没有嵌套，则可以按照任何顺序去评估。条件也可以通过IfAll、IfAny和IfNot组合复合条件。他们对于的是AND、OR、NOT。用户可以创建自定义条件或使用内置条件：IfFileName：接受路径（相对于基本路径）与正则表达式或glob匹配的文件。IfLastModified：接受与指定持续时间相同或更早的文件。IfAccumulatedFileCount ：在file tree walk期间超过一些计数阈值后接受路径。IfAccumulatedFileSize：在file tree walk期间超过累积文件大小阈值后接受路径。ifAll:如果所有嵌套条件都接受它（逻辑与），则接受路径。嵌套条件可以按任何顺序进行评估。IfAny:如果其中一个嵌套条件接受（OR或OR），则接受路径。嵌套条件可以按任何顺序进行评估。IfNot:如果嵌套条件不接受（逻辑NOT），则接受路径。 |
| scriptCondition | ScriptCondition | 如果没有指定PathConditions，则为必需。指定脚本的ScriptCondition元素。ScriptCondition应包含一个[Script，ScriptRef或ScriptFile](http://logging.apache.org/log4j/2.x/manual/appenders.html#ScriptCondition)元素，用于指定要执行的逻辑。有关配置[ScriptFiles](http://logging.apache.org/log4j/2.x/manual/filters.html#Script)和ScriptRefs的更多示例，请参阅ScriptFilter文档。该脚本传递了许多参数，包括在基本路径下找到的路径列表（最多为maxDepth），并且必须返回具有要删除路径的列表。 |

IfFileName 条件参数:

| **参数名称** | **类型** | **描述** |
| --- | --- | --- |
| glob | String | 如果regex没有指定的话，则必须。使用类似于正则表达式但是又具有更简单的有限模式语言来匹配相对路径（相对于基本路径） |
| regex | String | 如果glob没有指定的话，则必须。使用由Pattern类定义的正则表达式来匹配相对路径（相对于基本路径） |
| nestedConditions | PathCondition[] | 一组可选的嵌套[PathConditions](http://logging.apache.org/log4j/2.x/manual/appenders.html#DeletePathCondition)如果存在任何嵌套条件，则在它们删除文件之前，必须都接受。仅当外部条件接受文件（如果路径名称匹配）时，才会评估嵌套条件。 |

IfLastModified条件参数：

| **参数名称** | **类型** | **描述** |
| --- | --- | --- |
| age | String | 必须。指定持续时间[duration](http://logging.apache.org/log4j/2.x/log4j-core/apidocs/org/apache/logging/log4j/core/appender/rolling/action/Duration.html#parseCharSequence)。该条件接受比指定持续时间更早或更旧的文件。 |
| nestedConditions | PathCondition[] | 一组可选的嵌套[PathConditions](http://logging.apache.org/log4j/2.x/manual/appenders.html#DeletePathCondition)如果存在任何嵌套条件，则在它们删除文件之前，必须都接受。仅当外部条件接受文件（如果文件足够老）时，才会评估嵌套条件。 |

IfAccumulatedFileCount 条件参数

| **参数名称** | **类型** | **描述** |
| --- | --- | --- |
| exceeds | int | 必须。将要删除文件的计数阈值。也就是需要保留的文件数。 |
| nestedConditions | PathCondition[] | 一组可选的嵌套[PathConditions](http://logging.apache.org/log4j/2.x/manual/appenders.html#DeletePathCondition)如果存在任何嵌套条件，则在它们删除文件之前，必须都接受。仅当外部条件接受文件（如果超过阈值计数）时，才会评估嵌套条件。 |

IfAccumulatedFileSize 条件参数

| **参数名称** | **类型** | **描述** |
| --- | --- | --- |
| exceeds | String | 必须。将删除文件累计阀值的大小。大小可以指定字节。后缀可以是KB, MB or GB例如：20MB。也就是要保留最接近该值大小的文件。 |
| nestedConditions | PathCondition[] | 一组可选的嵌套[PathConditions](http://logging.apache.org/log4j/2.x/manual/appenders.html#DeletePathCondition)如果存在任何嵌套条件，则在它们删除文件之前，必须都接受。仅当外部条件接受文件（如果超过了阈值累积文件大小）时，才会评估嵌套条件。 |

下面这个示例配置使用的是RollingFileAppender，并且使用的是cron触发策略，是在每天的午夜触发。归档存储的目录是基于当前年和月。在rollover时间内匹配删除基本目录下所有满足参数glob等于\*/app-\*.log.gz和超过60天或更早的文件。

<?xml version="1.0" encoding="UTF-8"?> <Configuration status="warn" name="MyApp" packages=""> <Properties> <Property name="baseDir">logs</Property> </Properties> <Appenders> <RollingFile name="RollingFile" fileName="${baseDir}/app.log" filePattern="${baseDir}/$${date:yyyy-MM}/app-%d{yyyy-MM-dd}.log.gz"> <PatternLayout pattern="%d %p %c{1.} [%t] %m%n" /> <CronTriggeringPolicy schedule="0 0 0 \* \* ?"/> <DefaultRolloverStrategy> <Delete basePath="${baseDir}" maxDepth="2"> <IfFileName glob="\*/app-\*.log.gz" /> <IfLastModified age="60d" /> </Delete> </DefaultRolloverStrategy> </RollingFile> </Appenders> <Loggers> <Root level="error"> <AppenderRef ref="RollingFile"/> </Root> </Loggers> </Configuration>

下面的示例配置使用的是RollingFileAppender并且触发策略是基于时间和文件大小。在未来100天内会创建100个归档，这些归档存储的目录是基于当前年和月的，并且会以gzip方式进行压缩每个归档，而且rollover是每小时发生一次。这个配置也将会删除匹配\*/app-\*.log.gz和超过30天或更早的文件。但是会保留大小进100G或者最近的10个文件，先到则为准（whichever comes first.）。我个人理解应该是以文件创建时间为准。

<?xml version="1.0" encoding="UTF-8"?> <Configuration status="warn" name="MyApp" packages=""> <Properties> <Property name="baseDir">logs</Property> </Properties> <Appenders> <RollingFile name="RollingFile" fileName="${baseDir}/app.log" filePattern="${baseDir}/$${date:yyyy-MM}/app-%d{yyyy-MM-dd-HH}-%i.log.gz"> <PatternLayout pattern="%d %p %c{1.} [%t] %m%n" /> <Policies> <TimeBasedTriggeringPolicy /> <SizeBasedTriggeringPolicy size="250 MB"/> </Policies> <DefaultRolloverStrategy max="100"> <!-- Nested conditions: the inner condition is only evaluated on files for which the outer conditions are true. --> <Delete basePath="${baseDir}" maxDepth="2"> <IfFileName glob="\*/app-\*.log.gz"> <IfLastModified age="30d"> <IfAny> <IfAccumulatedFileSize exceeds="100 GB" /> <IfAccumulatedFileCount exceeds="10" /> </IfAny> </IfLastModified> </IfFileName> </Delete> </DefaultRolloverStrategy> </RollingFile> </Appenders> <Loggers> <Root level="error"> <AppenderRef ref="RollingFile"/> </Root> </Loggers> </Configuration>

ScriptCondition 参数：

| **参数名称** | **类型** | **描述** |
| --- | --- | --- |
| script | Script, ScriptFile or ScriptRef | Script元素是用于指定需要执行的逻辑。在基本路径下找到的路径列表，该脚本要是通过的，并且要以java.util.List<PathWithAttributes>返回删除的路径。 参考[ScriptFilter](http://logging.apache.org/log4j/2.x/manual/filters.html#Script)文档，如何配置ScriptFilter和ScriptRefs 的示例。 |

Script 参数：

| **参数名称** | **类型** | **描述** |
| --- | --- | --- |
| basePath | java.nio.file.Path | 删除操作开始扫描要删除的文件的目录。可用于相对路径列表中的路径。 |
| pathList | java.util.List | 在基本路径下找到路径列表直到指定的最大深度，优先排序最近修改的文件。该脚本可以自由修改并返回此列表。 |
| statusLogger | StatusLogger | StatusLogger通常用于在脚本执行期间记录内部事件。 |
| configuration | Configuration | 拥有此ScriptCondition的配置。 |
| substitutor | StrSubstitutor | StrSubstitutor用于替换查找变量。 |
| ? | String | 在配置中声明的任何属性 |

以下示例配置使用RollingFileAppender并且是cron触发策略，触发时间为每天午夜。归档存储目录是基于当前年和月的。该脚本将返回在基本目录下，日期为13号并且是星期五的rollover文件列表。删除操作将删除脚本返回的所有文件。

<?xml version="1.0" encoding="UTF-8"?> <Configuration status="trace" name="MyApp" packages=""> <Properties> <Property name="baseDir">logs</Property> </P